

Raport Rozproszone Systemy Informatyczne

WCF – serwis REST

Aleksandra Wolska
Szymon Łopuszyński
Konfiguracja dwumaszynowa

Windows Communication Foundation (WCF) to technologia umożliwiająca tworzenie usług sieciowych (services) w aplikacjach .NET, w tym usług REST (Representational State Transfer).

Usługi REST w WCF korzystają z architektury REST, która jest oparta na idei zasobów (np. obiektów danych, znajdujących się pod unikalnymi adresami URI (Uniform Resource Identifier) oraz na czterech podstawowych operacjach HTTP: GET, POST, PUT i DELETE. W przeciwieństwie do bardziej skomplikowanych protokołów usług sieciowych, takich jak SOAP, usługi REST są bezstanowe i korzystają z prostych formatów danych, takich jak JSON i XML. Bezstanowość polega na tym, że każde żądanie od klienta do serwera musi zawierać wszystkie informacje potrzebne do wykonania żądania. Serwer nie przechowuje żadnych informacji o stanie klienta pomiędzy żądaniami. Architektura REST zakłada podział na klienta i serwer, gdzie klient jest odpowiedzialny za interfejs użytkownika, a serwer za przechowywanie i przetwarzanie danych.

Tworzenie usługi REST w WCF obejmuje następujące kroki:

- Definiowanie interfejsu usługi: Interfejs definiuje metody, które będą dostępne dla konsumentów usługi.
- Implementacja usługi: Usługa jest implementowana jako klasa .NET, która implementuje zdefiniowany wcześniej interfejs.
- Konfiguracja usługi: Usługa musi być skonfigurowana do obsługi protokołu HTTP i formatu danych JSON lub XML.
- Hostowanie usługi: Usługa musi być uruchomiona na serwerze, aby była dostępna dla konsumentów.

Choć WCF umożliwia tworzenie usług REST, warto zauważyć, że technologia ta jest stopniowo zastępowana przez nowsze technologie, takie jak ASP.NET Core, które oferują większą elastyczność i wydajność. Wersja .NET 4.8 jest ostatnią wersją frameworka .NET, która obsługuje WCF.

1. Wymagania
 - a. Visual Studio 2022 z pakietem roboczym tworzenia aplikacji ASP.NET i aplikacji internetowych, w tym opcjonalny pakiet "Windows Community Foundation"
 - b. Dwie maszyny połączone znajdujące się w tej samej sieci udostępnianej przez hotspot, komputer hostujący posiada otwarte w zaporze porty 8080 i 10000
 - c. Import autorskiej biblioteki MyData do wyświetlania danych
2. Definiowanie aplikacji serwisu WCF z kontraktem
 - a. Tworzymy nowy projekt RestService
 - b. Wybieramy szablon "WCF Service Application"
 - c. W dalszej konfiguracji wybieramy wersję platformy .NET 4.8
 - d. zmieniamy nazwę istniejącego interfejsu IService na IRestService, w którym definiujemy kontrakt serwisu wraz z wszystkimi endpointami i ich formatami zapytań oraz odpowiedzi (Xml/Json), oraz kontraktowy typ danych Person osoby w naszej bazie.
3. Implementacja kontraktu
 - a. W pliku Service1.svc.cs zamieniamy implementację Service1 na RestService implementując nasz kontrakt.
 - b. W klasie RestService definiujemy metody z interfejsu do obsługi kolekcji użytkowników, oraz pole typu List<Person>, w której będziemy przechowywać dane użytkowników w pamięci programu, oraz statyczne pole _id będące licznikiem indeksów.
 - c. Specyfikujemy pojedynczą instancję serwisu dla wszystkich wywołań anotacją [ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]
 - d. Modyfikujemy plik Web.config aby obsłużyć zapytania REST. W sekcji <system.serviceModel> dodajemy opis naszej usługi, a następnie w sekcji <behaviors> specyfikujemy zachowanie naszych endpointów

```

<system.serviceModel>
  <services>
    <service name="MyWebService.RestService">
      <endpoint address="" binding="webHttpBinding"
contract="MyWebService.IRestService"
behaviorConfiguration="myRESTEndpointBehavior"> </endpoint>
    </service>
  </services>
  <behaviors>
    <endpointBehaviors>
      <behavior name="myRESTEndpointBehavior">
        <webHttp helpEnabled="true" />
      </behavior>
    </endpointBehaviors>
    ...
  </services>
</system.serviceModel>

```

Należy upewnić się, że flaga httpGetEnabled w sekcji serviceMetadata ma wartość "true".

4. Włączenie wsparcia dla zapytań CORS

- a. W projekcie tworzymy nowy plik "Global application class" Global.asax. który odpowiada za reagowanie na zdarzenia zgłoszone z poziomu aplikacji i poziomu sesji takie jak start aplikacji, zakończenie sesji, itp. dzięki metodom zaimplementowanym w klasie, reagującym na zdarzenia.
- b. W metodzie Application_BeginRequest, wywoływanej w przypadku otrzymania zapytania. W przypadku odpowiedniej metody dodajemy uprawnienia do headera zapytania

```
protected void Application_BeginRequest(object sender, EventArgs e)
{
```

```
    HttpContext.Current.Response.AddHeader("Access-Control-Allow-Origin", "*");
    if (HttpContext.Current.Request.HttpMethod == "OPTIONS")
```

```
    {
```

```
        HttpContext.Current.Response.AddHeader(
            "Access-Control-Allow-Methods",
            "POST, PUT, DELETE");
```

```
        HttpContext.Current.Response.AddHeader(
            "Access-Control-Allow-Headers",
            "Content-Type, Accept");
```

```
        HttpContext.Current.Response.AddHeader(
            "Access-Control-Max-Age", "7200");
        HttpContext.Current.Response.End();
```

```
    }
```

```
}
```

- c. W funkcji Application_Start umieszczamy wywołanie funkcji info() z klasy MyData, w celu uzyskania informacji przy uruchomieniu aplikacji.

```
protected void Application_Start(object sender, EventArgs e)
```

```
{
```

```
    MyData.MyData.info();
```

```
}
```

5. Konfiguracja środowiska dwumaszynowego

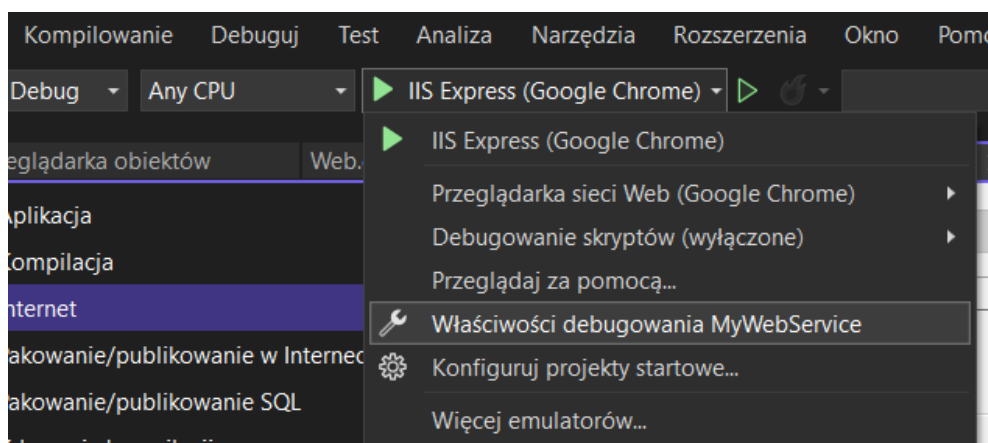
- a. Aplikacja serwisu hostowana jest na serwerze IIS Express
- b. W celu konfiguracji środowiska rozproszonego musimy otworzyć porty 8080 i 10000 w zaporze systemowej
- c. Następnie kompilujemy projekt i przechodzimy do pliku applicationhost.config w lokalizacji ...\\MyWebService\\.vs\\MyWebService\\config
- d. W pliku konfiguracyjnym wyszukujemy znacznik <binding> znajdujący się w sekcji <sites>
- e. Dla strony o nazwie takiej jak nazwa naszego projektu w sekcji <bindings> definiujemy dopuszczalne adresy na których serwer IIS może hostować usługę. W naszym przypadku na otwartym porcie 10000 z adresem ip4 komputera 192.168.43.18

```

<sites>
  <site name="WebSite1" id="1" serverAutoStart="true">
    <application path="/">
      <virtualDirectory path="/" physicalPath="%IIS_SITES_HOME%\WebSite1" />
    </application>
    <bindings>
      <binding protocol="http" bindingInformation=":8080:localhost" />
    </bindings>
  </site>
  <site name="MyWebService" id="2">
    <application path="/" applicationPool="Clr4IntegratedAppPool">
      <virtualDirectory path="/" physicalPath="D:\Studia\Sem6\Rozproszone systemy
informatyczne\Lab7\MyWebService\MyWebService" />
    </application>
    <bindings>
      <binding protocol="http" bindingInformation="*:10000:192.168.43.18" />
    </bindings>
  </site>
  <siteDefaults>
    <!-- To enable logging, please change the below attribute "enabled" to "true" -->
    <logFile logFormat="W3C" directory="%AppData%\Microsoft\IISExpressLogs"
enabled="false" />
    <traceFailedRequestsLogging directory="%AppData%\Microsoft" enabled="false"
maxLogFileSizeKB="1024" />
  </siteDefaults>
  <applicationDefaults applicationPool="Clr4IntegratedAppPool" />
  <virtualDirectoryDefaults allowSubDirConfig="true" />
</sites>

```

f. Ostatnim krokiem jest ustawienie adresu usługi w konfiguracji serwera IIS.



Wybieramy opcje jak na zdjęciu.

- g. Wybieramy zakładkę internet i w sekcji Serwery dla serwera IIS ustawiamy adres na wcześniej umieszczony w pliku konfiguracyjnym

The screenshot shows the IIS Manager console. On the left, the 'Internet' tab is selected in the navigation pane. The main area displays the 'Serwery' (Servers) section. The 'Akcja uruchamiania' (Start Action) section is expanded, showing options like 'Bieżąca strona' (Current page) and 'Początkowy adres URL' (Default URL). The 'Serwery' section is also expanded, showing the 'Zastosuj ustawienia serwera do wszystkich użytkowników' (Apply server settings to all users) checkbox, which is checked. Below this, the 'IIS Express' dropdown is selected, and the 'Liczba bitów' (Number of bits) is set to 'Domyślnie' (Default). The 'Adres URL projektu' (Project URL) field is set to 'http://192.168.43.18:10000/'. The 'Przesłoni korzeń adresu URL aplikacji' (Override application URL root) checkbox is unchecked, and the 'http://192.168.43.18:10000/' field is visible below it.

6. Implementacja obsługi klienta

- Tworzymy nowy projekt aplikacji konsolowej .NET
- Definiujemy klasę MyRestClient w której definiujemy adres naszego serwisu, oraz definiujemy metodę obsługującą zapytania

```
private static readonly string ADDRESS = "http://192.168.43.18:10000/Service1.svc/";  
public void processRequest(string endpoint, string method, string type)
```

```
{
```

```
    try
```

```
    {
```

```
        HttpWebRequest req = WebRequest.Create(ADDRESS + endpoint) as
```

```
HttpWebRequest;
```

```
        req.KeepAlive = false;
```

```
        req.Method = method;
```

```
        req.ContentType = type;
```

```
        string payload = "";
```

```
        if(method == "POST" || method == "PUT") {
```

```
            if (type == "application/json")
```

```
            {
```

```
                payload = getJsonPerson();
```

```
            }
```

```
        else
```

```
        {
```

```

        payload = getXmlPerson();
    }
}
switch(method)
{
    case "GET":
        break;
    case "POST":
        byte[] buforPost = Encoding.UTF8.GetBytes(payload);
        req.ContentLength = buforPost.Length;
        Stream postData = req.GetRequestStream();
        postData.Write(buforPost, 0, buforPost.Length);
        postData.Close();
        break;
    case "PUT":
        byte[] buforPut = Encoding.UTF8.GetBytes(payload);
        req.ContentLength = buforPut.Length;
        Stream putData = req.GetRequestStream();
        putData.Write(buforPut, 0, buforPut.Length);
        putData.Close();
        break;
    case "DELETE":
        break;
}

HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
Encoding enc = Encoding.GetEncoding(1252);
StreamReader responseStream = new StreamReader(resp.GetResponseStream(),
enc);
string responseString = responseStream.ReadToEnd();
responseStream.Close();
resp.Close();
Console.WriteLine(responseString);
}
catch(Exception ex)
{
    Console.WriteLine(ex.Message.ToString());
}
}

```

c. Działanie obsługi zapytań :

Na początku procesu tworzymy obiekt `HttpRequest` używając adresu URL usługi RESTful.

```
HttpRequest req = HttpRequest.Create(ADDRESS + endpoint) as
HttpRequest;
```

Następnie ustalamy metodę HTTP dla żądania - może to być GET, POST, PUT lub DELETE. Metoda jest wybierana na podstawie parametru przekazanego do funkcji `processRequest`.

```
req.Method = method;
```

Dodajemy nagłówek "Content-Type" do żądania, wskazując na to, czy dane są przesyłane w formacie JSON czy XML.

```
req.ContentType = type;
```

Dla żądań typu POST i PUT, tworzymy ładunek (payload) żądania w formacie JSON lub XML, w zależności od określonego nagłówka Content-Type.

```
string payload = "";
if(method == "POST" || method == "PUT") {
    if (type == "application/json")
    {
        payload = getJsonPerson();
    }
    else
    {
        payload = getXmlPerson();
    }
}
```

Wysyłamy żądanie do serwera. W przypadku żądań typu POST i PUT, dołączamy do żądania ładunek, który został wcześniej utworzony.

```
byte[] buforPut = Encoding.UTF8.GetBytes(payload);
req.ContentLength = buforPut.Length;
Stream putData = req.GetRequestStream();
putData.Write(buforPut, 0, buforPut.Length);
putData.Close();
```

Po wysłaniu żądania, odczytujemy odpowiedź z serwera.

```
HttpWebResponse resp = req.GetResponse() as HttpWebResponse;
StreamReader responseStream = new
StreamReader(resp.GetResponseStream(), enc);
```

```
string responseString = responseStream.ReadToEnd();
```

Dodatkowo definiujemy funkcje opakowujace odebrane od uzytkownika dane osoby w odpowiedni format Xml/Json getXmlPerson() i getJsonPerson()

- d. W pliku Program.cs definiujemy menu uzytkownika pokrywajace zapytania dla wszystkich udostepnionych endpointow i umozliwiajaca wybor spośród nich uzytkownikowi, po czym zwracamy odpowiedzi serwera na ekran, lub obsluguje wyrzucone wyjatki.

7. Działanie programu

Klient: Uruchomienie i wylistowanie ludzi XML

```
C:\Users\Aleksandra\Desktop\Polibuda\RSI\REPO\RSI\Lab7\RestClient\Re...
Aleksandra Wolska, 251810
Szymon Łopuszyński, 260454
9 maja, 09:29:50
4.0.30319.42000
Aleksandra
Microsoft Windows NT 6.2.9200.0
192.168.43.194

-----
Wybierz operacje:
-----

XML
  1. Wszyscy ludzie
  2. Jedna osoba
  3. Nowa osoba
  4. Usuń osobę
  5. Edytuj osobę
  6. Liczba ludzi

JSON
  7. Wszyscy ludzie
  8. Jedna osoba
  9. Nowa osoba
  10. Usuń osobę
  11. Edytuj osobę
  12. Liczba ludzi
0. Exit
1
<ArrayOfPerson xmlns="http://schemas.datacontract.org/2004/07/MyWebService"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><Person><Id>0</Id><Name>
Aleksandra Wolska</Name><Age>23</Age><Email>wschodzaca@gmail.com</Email><Pe
rson><Person><Id>1</Id><Name>szymon A?opuszyA"ski</Name><Age>22</Age><Email>
szym3k@op.pl</Email></Person><Person><Id>2</Id><Name>Jan Kowalski</Name><Age
>13</Age><Email>kowal@wp.pl</Email></Person><Person><Id>3</Id><Name>Ola2 Wol
ska2</Name><Age>23</Age><Email>ola2@wp.pl</Email></Person></ArrayOfPerson>
```

Klient: Wylistowanie ludzi JSON

```
-----
Wybierz operacje:
-----

XML
  1. Wszyscy ludzie
  2. Jedna osoba
  3. Nowa osoba
  4. Usuń osobę
  5. Edytuj osobę
  6. Liczba ludzi

JSON
  7. Wszyscy ludzie
  8. Jedna osoba
  9. Nowa osoba
  10. Usuń osobę
  11. Edytuj osobę
  12. Liczba ludzi
0. Exit
7
[{"Id":0,"Name":"Aleksandra Wolska","Age":23,"Email":"wschodzaca@gmail.com"},
{"Id":1,"Name":"szymon A?opuszyA"ski","Age":22,"Email":"szym3k@op.pl"},{"Id
":2,"Name":"Jan Kowalski","Age":13,"Email":"kowal@wp.pl"},{"Id":3,"Name":"Ol
a2 Wolska2","Age":23,"Email":"ola2@wp.pl"}]
```


Klient: Wypisanie ilości ludzi

```
-----
Wybierz operacje:
-----

XML
 1. Wszyscy ludzie
 2. Jedna osoba
 3. Nowa osoba
 4. Usuń osobę
 5. Edytuj osobę
 6. Liczba ludzi

JSON
 7. Wszyscy ludzie
 8. Jedna osoba
 9. Nowa osoba
10. Usuń osobę
11. Edytuj osobę
12. Liczba ludzi

0. Exit
6
<int xmlns="http://schemas.microsoft.com/2003/10/Serialization/">4</int>

-----
Wybierz operacje:
-----

XML
 1. Wszyscy ludzie
 2. Jedna osoba
 3. Nowa osoba
 4. Usuń osobę
 5. Edytuj osobę
 6. Liczba ludzi

JSON
 7. Wszyscy ludzie
 8. Jedna osoba
 9. Nowa osoba
10. Usuń osobę
11. Edytuj osobę
12. Liczba ludzi

0. Exit
12
<int xmlns="http://schemas.microsoft.com/2003/10/Serialization/">4</int>
```

Klient: Dodanie osoby

```
-----
Wybierz operacje:
-----

XML
 1. Wszyscy ludzie
 2. Jedna osoba
 3. Nowa osoba
 4. Usuń osobę
 5. Edytuj osobę
 6. Liczba ludzi

JSON
 7. Wszyscy ludzie
 8. Jedna osoba
 9. Nowa osoba
10. Usuń osobę
11. Edytuj osobę
12. Liczba ludzi

0. Exit
3
Podaj imie i nazwisko osoby: Ewa Ewacka
Podaj wiek osoby: 21
Podaj email: ewa@wp.pl
<Person xmlns="http://schemas.datacontract.org/2004/07/MyWebService" xmlns:i
="http://www.w3.org/2001/XMLSchema-instance"><Id>0</Id><Name>Ewa Ewacka</Nam
e><Age>21</Age><Email>ewa@wp.pl</Email></Person>
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">Added ne
w person: 5 Ewa Ewacka age: 21 ewa@wp.pl</string>
```

Klient: Edycja osoby

Wybierz operacje:

- XML
1. Wszyscy ludzie
 2. Jedna osoba
 3. Nowa osoba
 4. Usuń osobę
 5. Edytuj osobę
 6. Liczba ludzi

- JSON
7. Wszyscy ludzie
 8. Jedna osoba
 9. Nowa osoba
 10. Usuń osobę
 11. Edytuj osobę
 12. Liczba ludzi

0. Exit

5

Podaj indeks osoby: 5

Podaj imię i nazwisko osoby: Karolina Karolska

Podaj wiek osoby: 23

Podaj email: karolina@wp.pl

```
<Person xmlns="http://schemas.datacontract.org/2004/07/MyWebService" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><Id>0</Id><Name>Karolina Karolska</Name><Age>23</Age><Email>karolina@wp.pl</Email></Person>
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">Edited person: 0 Karolina Karolska age: 23 karolina@wp.pl</string>
```

Klient: Próba usunięcia nieistniejącej osoby

Wybierz operacje:

- XML
1. Wszyscy ludzie
 2. Jedna osoba
 3. Nowa osoba
 4. Usuń osobę
 5. Edytuj osobę
 6. Liczba ludzi

JSON

7. Wszyscy ludzie
8. Jedna osoba
9. Nowa osoba
10. Usuń osobę
11. Edytuj osobę
12. Liczba ludzi

0. Exit

4

Podaj indeks osoby: 10

The remote server returned an error: (404) Not Found.

Klient: prezentacja inkrementacji indeksu dla pustej bazy

```
-----
Wybierz operacje:
-----

XML
1. Wszyscy ludzie
2. Jedna osoba
3. Nowa osoba
4. Usuń osobę
5. Edytuj osobę
6. Liczba ludzi

JSON
7. Wszyscy ludzie
8. Jedna osoba
9. Nowa osoba
10. Usuń osobę
11. Edytuj osobę
12. Liczba ludzi
0. Exit
1
<ArrayOfPerson xmlns="http://schemas.datacontract.org/2004/07/MyWebService"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance"/>
-----
Wybierz operacje:
-----

XML
1. Wszyscy ludzie
2. Jedna osoba
3. Nowa osoba
4. Usuń osobę
5. Edytuj osobę
6. Liczba ludzi

JSON
7. Wszyscy ludzie
8. Jedna osoba
9. Nowa osoba
10. Usuń osobę
11. Edytuj osobę
12. Liczba ludzi
0. Exit
3
Podaj imie i nazwisko osoby: Jan Janowski
Podaj wiek osoby: 30
Podaj email: jan@wp.pl
<Person xmlns="http://schemas.datacontract.org/2004/07/MyWebService" xmlns:i
="http://www.w3.org/2001/XMLSchema-instance"><Id>0</Id><Name>Jan Janowski</N
ame><Age>30</Age><Email>jan@wp.pl</Email></Person>
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">Added ne
w person: 7 Jan Janowski age: 30 jan@wp.pl</string>
```

Zrzut ekranu z hosta serwisu

```
Aleksandra Wolska, 251810
Szymon kopuszyński, 260454
25 kwietnia, 09:47:55
4.0.30319.42000
shibe
Microsoft Windows NT 6.2.9200.0
192.168.43.18
00:00:30
--> Endpointy:
Service endpoint: BasicHttpBinding_IDatabaseService
Binding: System.ServiceModel.BasicHttpBinding
ListenUri: http://192.168.43.18:10000/DatabaseService/endpoint1
Service endpoint: WSHttBinding_IDatabaseService
Binding: System.ServiceModel.WSHttBinding
ListenUri: http://192.168.43.18:10000/DatabaseService/endpoint2
Service is started and running.
Press <ENTER> to STOP service...

All users
Remove user Id: 0 Aleksandra Wolska, age: 22
All users
Update user Id: 1 Szymek Szymański, age: 21
All users
Add user Id: 3 Kasia Kowalska, age: 23
All users
All user async start
All user async end
All users
All users
Add user Id: 3 Ala Alowska, age: 25
All users
All user async start
All user async end
```

Źródła:

- Instrukcja Dr. Frasia - Ćwiczenie 4 - wersja WCF
- film pokazujący zmianę adresu ip serwera IIS :
https://www.youtube.com/watch?v=_t9u9DKIKP4&t=154s&ab_channel=ITProGuide
- Wykład dr. Frasia